first element 420 to second element 430. Link 440 remains visible to all processes other than the insertion operation at the conclusion of the update phase of step 130. In this manner, the integrity of data structure 240 is maintained. A new link 450 points from first element 420 to new element 410. New link 450 is visible to the insertion operation during the update phase of step 130 and is not visible to other operations until the commit phase of step 140 (FIG. 1). New element 410 (FIG. 4) also includes a link 460 pointing to second element 430.

[0032] FIG. 4B illustrates the result of a state transition from the pending insert state to a valid state at the conclusion of the commit phase of step 140 (FIG. 1). During the commit phase computer code 220 removes link 440 and makes new links 450 and 460 visible to all processes. Both of these operations are executed atomically. In the preferred embodiment all operations executed during the commit phase of step 140 are atomic operations.

[0033] FIGs. 5 and 6 further illustrate the process steps for inserting new element 410 (FIG. 4) into data structure 240 (FIG. 2) according to one embodiment of the invention. With reference to FIG. 5, the process steps of the insertion operation which occur during the update phase of step 140 (Fig. 1) are illustrated. In accordance with the process of FIG. 1, in a step 510 computer code 220 receives operations operable upon

data structure 240 from task queue 260. In this example the first operation in task queue 260 is an insertion operation of new element 410.

[0034] In a step 520 the update phase of step 130 starts. In a step 530 new element 410 to be inserted by the insertion operation is created by computer code 220 if new element 410 does not already exist and a field 247 of the new element 410 is modified to indicate that new element 410 is in the pre-associated state.

[0035] In a step 540 field 247 within new element 410 is modified to indicate that new element 410 has transitioned from the pre-associated state to the pending insert state.

[0036] In a step 550 computer code 220 navigates through data structure 240 to first element 420 which will become a parent of new element 410 if the insertion operation executes successfully. Step 550 further includes the execution of tasks associated with the insertion of new element 410 such as creating links 450 and 460 and a pointer to the insertion point for use during the commit phase of step 140 (FIG. 1).

[0037] In a decision step 560, computer code 220 determines if there are existing update data 250 associated with the row 320 in which link 450 to new element 410 is being inserted. If no such update data 250 yet exists, then, in a step 570, memory for

update data 250 is allocated and pointed to by PU 330 as described herein and illustrated in FIG. 3.

[0038] In a step 580 instructions for transitioning the state of first element 420 from the pending insert state to the valid state during the commit phase of step 140 (FIG. 1) are written to (pushed on) update data 250 and become top item 340. The written instructions include a flag (not shown) indicating that conflict tests have not been checked for the insertion operation.

[0039] Returning to decision step 560, if update data 250 associated with the row in which link 450 to new element 410 to be inserted already exists, then computer code 220 executes a step 582 identical to step 580. After writing data to update data 250 computer code 220 traverses update data 250 in a step 584, testing for operations that would conflict with the insertion of new element 410. Methods of identifying conflicting operations include those described in co-pending application entitled "System and Method for Determining the Commutativity of Computational Operations."

[0040] In a decision step 586 computer code 220 considers if conflicting operations were found in step 584. If a conflicting operation is identified then the insertion operation is not executable and the insertion is blocked from completion in a step 588. The blocking step 588 optionally includes postponing